# Conversión y adaptación de documentos XML

# 2.- Estructura básica de una hoja XSLT.

Se indica así:

<?xml-stylesheet type="text/xsl" href="ruta\_del\_fichero\_xsl.xsl"?>

# 3. Elementos XSLT

El elemento raíz de una hoja XSLT es xsl:stylesheet o xls:transform, que son prácticamente equivalentes. Sus atributos principales son:

* version, cuyo valor puede ser 1.0 o 2.0.
* xmlns:xsl, se utiliza para declarar el espacio de nombres xsl. Para XSLT suele ser la dirección: <http://www.w3.org/1999/XSL/Transform>

# 4.Xpath

XPath es un estándar (diferente de XML) aprobado por el W3C, que permite navegar entre los elementos y atributos de un documento XML.

<?xml version="1.0" encoding="UTF-8"?>

<matriculas>

  <alumno>

    <nombre>Pedro</nombre>

    <apellido1>López</apellido1>

    <apellido2>Ortega</apellido2>

    <DNI pais="es">López</DNI>

  </alumno>

</matriculas>

Al evaluar la expresión "/matriculas/alumno/nombre" podríamos obtener los nombres de los alumnos matriculados.

## 4.1.- Términos básicos.

### Nodos

* Nodo raíz, es el nodo que contiene al ejemplar del fichero XML.
  + ¡Cuidado! No lo confundamos con el elemento raíz del documento, ya que éste último está por debajo de él.
* Nodos elemento, son cada uno de los elementos del documento XML.
  + Tienen un elemento padre.
  + El padre del elemento raíz, es decir del ejemplar, es el nodo raíz del documento.
  + Pueden tener identificadores únicos, lo que permite referenciarlos de forma mucho más directa. Para ello es necesario que el atributo esté definido en un DTD o un fichero XSD asociado.
* Nodos texto, son aquellos caracteres del documento que no están marcados con ninguna etiqueta.
  + No tienen hijos.
* Nodos atributo, son los atributos de un elemento.
  + Se consideran etiquetas añadidas al nodo elemento.
  + No se consideran hijos de ese elemento.
  + Aquellos atributos que tengan un valor asignado en el esquema asociado, se tratarán como si ese valor se le hubiese dado al escribir el documento XML.
  + Para las definiciones de los espacios de nombre y para aquellos atributos que se han definido con la propiedad #IMPLIED en su DTD no se crean nodos.

<?xml version="1.0" encoding="UTF-8"?>

<matriculas>

  <alumno>

    <nombre>Pedro</nombre>

    <apellido1>López</apellido1>

    <apellido2>Ortega</apellido2>

    <DNI pais="es">López</DNI>

  </alumno>

</matriculas>

* <matrículas> es el nodo elemento raíz
* <nombre>Pedro</nombre> es un nodo elemento
* pais="es" es un nodo atributo

### Items

Los ítems pueden ser nodos o valores atómicos.

En el ejemplo, valores atómicos serían "Pedro", o "es"

### Relaciones entre nodos

Según como si sitúen los nodos dentro del árbol, se habla de las relaciones entre ellos. Así se dice que hay relaciones padre-hijo, hermano, ascendentes y descendentes.

En ejemplo anterior, algunas de estas relaciones son:

* <nombre> es hijo de <alumno>.
* <nombre> y <DNI> son hermanos.
* <matriculas> es un ascendiente de <nombre>
* <apellido1> es descendiente de <matriculas>

## 4.2.- Expresiones.

### Resultados que da la evaluación de una expresión XPath

* Un conjunto de nodos (node-set)
  + No está ordenado.
  + Se considera que todos los elementos de un conjunto de nodos son hermanos, independientemente de lo que fuesen originalmente.
  + Aunque los hijos de los nodos que forman un conjunto de nodos son accesibles, los subárboles de un nodo no se consideran elementos del conjunto.
* Un valor booleano.
* Un número.
* Una cadena.

### ¿Qué elementos podemos utilizar en una expresión XPath?

* Agrupaciones: “()”, “{}”, “[ ]”.
* Elemento actual, elemento padre.
* Atributos: “@”.
* Elementos, “\*”.
* Separadores, “::”.
* Comas, “,”.
* El nombre de un elemento.
* Tipo de nodo, que puede ser:
  + comment.
  + text.
  + procesing instruction.
  + node.
* Operadores: and, or, mod, div, \*, /, //, |, +, -, =, !=, <, >, <=, >=.
* Nombres de función.
* Denominación de ejes: ancestor, ancestor-or-self-atribute, child, descendant, descendant-or-self, following, following-sibling, namespace, parent, preceding, preceding-sibling, self.
* Literales, se ponen entre comillas dobles o simples. Pueden anidarse alternando el tipo de comillas.
* Números.
* Referencias a variables, para lo que se utiliza la sintaxis: $nombreVariable.

## 4.3.- Procesando expresiones.

* Nodo actual, es aquél en el que se encuentra el procesador.
* Nodo contexto, cada expresión está formada por subexpresiones que se van evaluando antes de resolver la siguiente. El conjunto de nodos obtenido tras evaluar una expresión y que se utiliza para evaluar la siguiente es el nuevo contexto.
* Tamaño del contexto, es el número de nodos que se están evaluando en un momento dado en una expresión XPath.

## 4.4.- Ruta de Localización.

La ruta de localización de un nodo es la ruta que hay que seguir en un árbol de datos para localizar ese nodo. Las rutas de localización se evalúan y esa evaluación siempre devuelve un conjunto de nodos, aunque puede estar vacío.

Las rutas de localización básicas son:

* La ruta de localización del nodo raíz del documento. Es la barra diagonal “/”. Se trata de una ruta absoluta, porque siempre significa lo mismo, independientemente de la posición del procesador en el documento de entrada al aplicar la regla.
* Localización de un elemento, selecciona todos los hijos del nodo de contexto con el nombre especificado. Los elementos a los que se refiera dependerán de la localización del nodo de contexto, por lo que es una ruta relativa. En el caso de que el nodo contexto no tenga ningún nodo hijo con esa denominación, el valor dela ruta de localización será un elemento vacío.
* Localización de atributos, para referirnos a ellos en XPath, se utiliza el símbolo “@” seguido del nombre del atributo deseado.
* Localización de espacios de nombres, no se tratan explícitamente.
* Localización de comentarios.
* Localización de nodos de texto.
* Localización de instrucciones de procesamiento.

### comparar distintos elementos y tipos de nodo simultáneamente

* Asterisco “\*”: compara cualquier nodo de elemento, independientemente de su nombre. No compara ni atributos ni comentarios, nodos de texto o instrucciones de procesamiento.
* Node(): compara, además de los tipos de elementos, el nodo raíz, los nodos de texto, los de instrucción de procesamiento, los nodos de espacio de nombre, los de atributos y los de comentarios.
* “@\*”: compara los nodos de atributo.

## 4.5.- Funciones de XPath.

Entre las funciones más importantes que podemos utilizar en XPath destacan:

* boolean(), al aplicarla sobre un conjunto de nodos devuelve true si no es vacío.
* not(), al aplicarla sobre un predicado devuelve true si el predicado es falso , y falso si el predicado es true.
* true(), devuelve el valor true.
* false(), devuelve el valor false.
* count(), devuelve el número de nodos que forman un conjunto de nodos.
* name(), devuelve un nombre de un nodo.
* local-name(), devuelve el nombre del nodo actual o del primer nodo de un conjunto de nodos.
* namespace-uri(), devuelve el URI del nodo actual o del primer nodo de un conjunto dado.
* position(), devuelve la posición de un nodo en su contexto comenzando en 1. Por ejemplo, para seleccionar los dos primeros elementos de tipo elemento de un fichero XML pondremos: //elemento[position()<=2]
* last(), Devuelve el último elemento del conjunto dado.
* normalize-space(), permite normalizar los espacios de una cadena de texto, es decir, si se introduce una cadena donde hay varios espacios consecutivos, esta función lo sustituye por uno solo.
* string(), es una función que convierte un objeto en una cadena. Los valores numéricos se convierten en la cadena que los representa teniendo en cuenta que los positivos pierden el signo. Los valores booleanos se convierten en la cadena que representa su valor, esto es “true” o “false”.
* concat(), devuelve dos cadenas de texto concatenadas. El ejemplo siguiente devuelve “XPath permite obtener datos de un fichero XML”.
* <matrículas> es el nodo elemento raíz
* <nombre>Pedro</nombre> es un nodo elemento
* pais="es" es un nodo atributo

## 4.6.- Predicados.

Un predicado es una expresión booleana que añade un nivel de verificación al paso de localización.

En estas expresiones podemos incorporar funciones XPath.

Lo que podemos incluir en un predicado.

* Ejes, permiten seleccionar el subárbol dentro del nodo contexto que cumple un patrón. Pueden ser o no de contenido.
  + child, es el eje utilizado por defecto. Su forma habitual es la barra, “/”, aunque también puede ponerse: /child::
  + attribute, permite seleccionar los atributos que deseemos. Es el único eje que veremos que no es de contenido.
  + descendant, permite seleccionar todos los nodos que descienden del conjunto de nodos contextos. Se corresponde con la doble barra, “//”, aunque se puede usar: descendant::
  + self, se refiere al nodo contexto y se corresponde con el punto ”.”.
  + parent, selecciona los nodos padre, para referirnos a él usamos los dos puntos, ”..”.
  + ancestor, devuelve todos los nodos de los que el nodo contexto es descendiente.
* Nodos test, permiten restringir lo que devuelve una expresión XPath. Podemos agruparlos en función de los ejes a los que se puede aplicar.
  + Aplicable a cualquier eje:
    - “\*”, solo devuelve elementos, atributos o espacios de nombres pero no permite obtener nodos de texto, o comentarios de cualquier tipo.
    - node(), devuelve los nodos de todos los tipos.
  + Solo aplicables a ejes de contenido:
    - text(), devuelve cualquier nodo de tipo texto.
    - comment(), devuelve cualquier nodo de tipo comentario.
    - processing-instruction(), devuelven cualquier tipo de instrucción de proceso.

## 4.9.- Acceso a datos de otro documento XML.

Además de acceder a los datos del fichero XML con el que se está trabajando directamente, es útil poder acceder a los datos de otros ficheros.

Para ello utilizaremos la función document(), pero dicha función NO es del lenguaje XPath, sino que pertenece a XSLT.

dos argumentos diferentes:

* document(URI)

En este caso la función devuelve el elemento raíz del documento XML que se localiza en el URI especificado.

* document(nodo)

# 5.- Utilización de plantillas.

El elemento xml:template

* atributo denominado match, que se utiliza para seleccionar los nodos del árbol de entrada (conforme a XPath) a los que se va a aplicar la plantilla.
* Para especificar el formato de salida, se emplean sentencias **XHTML**
* Para especificar el punto de la salida donde queremos que se apliquen las plantillas, se emplea el elemento xsl:apply-templates.
* Tiene un atributo select, que se utiliza para seleccionar los hijos del nodo de entrada (conforme a XPath) a los que se va a aplicar la plantilla.
* Podemos usar ese atributo para especificar el orden en que van a ser procesados los nodos hijo. Si no lo hacemos de este modo, se aplicarán en el orden es el que el interprete utiliza al leer el documento XML, es decir, de arriba abajo.

# 6.- Procesadores XSLT.

Un procesador XSLT es un software que lee un documento XSLT y otro XML, y crea un documento de salida aplicando las instrucciones de la hoja de estilos XSLT a la información del documento XML.

Pueden estar integrados dentro de un explorador Web, en un servidor web, o puede ser un programa que se ejecuta desde la línea de comandos.

* Existen diferentes modos de realizar la transformación XSLT
* Mediante el procesador MSXML, (servicios principales de Microsoft XML).
* Usando un procesador XSLTPROC, por ejemplo xsltproc desde línea de comandos.
* Invocando a la biblioteca de transformación desde un programa.
* Realizando un enlace entre la hoja XSLT y el documento XML, en este caso hay que añadir, en el fichero XML entre el la definición de la versión XML y la definición del tipo de documento, la línea:

<?xml-stylesheet type=”text/xsl” href=”path\_hoja\_xsl”?>

# 7.- Depuradores XSLT.

Los depuradores son elementos de software que permiten seguir la generación de un documento, a partir de los datos de un documento XML aplicándoles una hoja de estilo XSLT.

Los depuradores XSLT nos facilitan la localización y corrección de errores dejándonos ejecutar el código paso a paso, salir, ejecutar el código hasta el cursor, ejecutar hasta el final, pausar y detener.

Algunos editores de XML (sobretodo los comerciales), incluyen un depurador que permite visualizar simultáneamente la plantilla que se está ejecutando y sobre qué datos del documento XML actúan y cuál es la salida que genera dicha orden.

De este modo es más fácil averiguar cuáles son las plantillas que dan lugar al formato que deseamos en la salida del documento.

# Ejercicios resueltos

## Ejercicio 1

<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>

<!DOCTYPE agenda>

<agenda>

  <propietario>

    <identificadores>

      <nombre>Alma</nombre>

      <apellidos>López Terán</apellidos>

    </identificadores>

    <direccion>

      <calle>El Percebe 13, 6F</calle>

      <localidad>Torrelavega</localidad>

      <cp>39300</cp>

    </direccion>

    <telefonos>

      <movil>970898765</movil>

      <casa>942124567</casa>

      <trabajo>628983456</trabajo>

    </telefonos>

  </propietario>

  <contactos>

    <persona id="p01">

      <identificadores>

        <nombre>Inés</nombre>

        <apellidos>López Pérez</apellidos>

      </identificadores>

      <direccion>

        <calle>El Ranchito 24, 6B</calle>

        <localidad>Santander</localidad>

        <cp>39006</cp>

      </direccion>

      <telefonos>

        <movil>970123123</movil>

      </telefonos>

    </persona>

    <persona id="p02">

      <identificadores>

        <nombre>Roberto</nombre>

        <apellidos>Gutiérrez Gómez</apellidos>

      </identificadores>

      <direccion>

        <calle>El Marranito 4, 2F</calle>

        <localidad>Santander</localidad>

        <cp>39004</cp>

      </direccion>

      <telefonos>

        <movil>970987456</movil>

        <casa>942333323</casa>

      </telefonos>

    </persona>

    <persona id="p03">

      <identificadores>

        <nombre>Juan</nombre>

        <apellidos>Sánchez Martínez</apellidos>

      </identificadores>

      <direccion>

        <calle>El Cangrejo 10, sn</calle>

        <localidad>Torrelavega</localidad>

        <cp>39300</cp>

      </direccion>

      <telefonos>

        <movil>997564343</movil>

        <casa>942987974</casa>

        <trabajo>677899234</trabajo>

      </telefonos>

    </persona>

  </contactos>

</agenda>

Construir las sentencias XPath que permitan obtener los siguientes datos:

* Nombre del propietario de la agenda.
  + /agenda/propietario/identificadores/nombre
* Teléfono de casa del propietario.
  + /agenda/propietario/telefonos/casa
* Nombres y apellidos de los contactos de la agenda.
  + //contactos/persona/identificadores/nombre | //contactos/persona/identificadores/apellidos
* Nombre e identificador de cada contacto.
  + //contactos/persona/identificadores/nombre | //contactos/persona/@id
* Datos del contacto con identificador "p02".
  + //contactos/persona[@id="p02"]/\*/\*
* Identificadores de los contactos que tienen teléfono en casa.
  + [//contactos/persona/telefonos/casa/../../@id](mailto://contactos/persona/telefonos/casa/../../@id)

## ejercicio 2

Dada la siguiente plantilla XSLT

<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">

<xsl:template match="identificadores">

<xsl:value-of select="nombre"/>,

<xsl:value-of select="apellidos"/>

</xsl:template>

<xsl:template match="persona">

<xsl:apply-templates select="identificadores"></xsl:apply-templates>

</xsl:template>

</xsl:stylesheet>

Que se desea aplicar sobre el siguiente fichero

<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>

<!DOCTYPE agenda>

<?xml-stylesheet type="text/xsl" href="./LMSGI\_CONT\_Ejemplo02.xsl"?>

<agenda>

<persona id="p01">

<identificadores>

<nombre>Inés</nombre>

<apellidos>López Pérez</apellidos>

</identificadores>

<direccion>

<calle>El Ranchito 24, 6B</calle>

<localidad>Santander</localidad>

<cp>39006</cp>

</direccion>

<telefonos>

<movil>970123123</movil>

</telefonos>

</persona>

<persona id="p02">

<identificadores>

<nombre>Roberto</nombre>

<apellidos>Gutiérrez Gómez</apellidos>

</identificadores>

<direccion>

<calle>El Marranito 4, 2F</calle>

<localidad>Santander</localidad>

<cp>39004</cp>

</direccion>

<telefonos>

<movil>970987456</movil>

<casa>942333323</casa>

</telefonos>

</persona>

</agenda>XML

Indicar cuál sería el resultado de aplicar la transformación XSLT sobre el fichero XML.

<?xml version="1.0" encoding="utf-8"?>

Inés,

López Pérez

Roberto,

Gutiérrez Gómez

Juan,

Sánchez Martínez

# Observaciones

1. Editores

Para evaluar expresiones Xpath podéis usar XML Copy Editor, que sí tenemos instalado en el

instituto.

Os dejo un vídeo de ejemplo con este editor para evaluar expresiones con XPath:

<https://www.youtube.com/watch?v=JT6DvaHkgPU>

Y otros 2 para utilizar con XSLT:

<https://www.youtube.com/watch?v=BF-rimBMzi4>

<https://youtu.be/zfxYneU1444>

Al final de todo, en los contenidos de la plataforma, tenéis un vídeo que os muestra cómo aplicar

transformaciones XML con el editor

<https://www.oxygenxml.com/xml_editor/download_oxygenxml_editor.html>

Tiene una versión gratuita de 30 días, tendréis que registraros. Es muy completo y muy cómodo,

pero para nuestro propósito, nos bastará con XML Copy Editor. Podéis usarlo como respaldo en caso

de que no consigáis que algo funcione o por mera curiosidad.

# Tests

1. Para lograr ir directamente al último de los elementos hijos de un elemento en el que nos encontramos habrá que usar la función:

* position(end)
* last()
* first()
* end()

2. Indica cuál de estas afirmaciones es correcta:

* La función document() devuelve únicamente una rama de un nodo dado.
* La función document() devuelve únicamente una rama de un nodo dado.
* No existe ningún modo de acceder a los elementos de varios documentos.
* XPath no proporciona un modo de acceder a datos de varios documentos simultáneamente.

3. El único modo de generar un documento a partir de una transformación XSLT de otro es utilizando un editor XML que tenga incorporado un procesador XSLT:

* No
* SI

4. Los lenguajes de marcas que no permiten la transformación de documentos son:

* HTML.
* XHTML.
* XPath.
* XSL

5. Cuáles de los siguientes tipos de documentos pueden obtenerse a partir de la transformación XSL de un documento XML:

* DOC.
* HTML.
* PDF.
* Texto.

6. Selecciona los elementos que tienen esta funcionalidad. Debes elegir entre uno de estos:

1. preserve-space
2. value-of
3. strip-space
4. decimal-format

Debes introducir sólo el número de la opción seleccionada en cada caso. Ej: 1

* Marca aquellos elementos que tienen espacios blancos eliminados antes de la transformación:
* Resuelve una expresión XPath dada:
* Marca los elementos que no tienen blancos eliminados:
* Convierte datos numéricos en cadenas de texto:

7. XPath es un lenguaje XML que permite:

* Transformar el formato de los datos de un fichero XML.
* Definir un vocabulario que ha de cumplir un documento XML.
* Obtener los datos del fichero XML de una base de datos.
* Acceder a los datos de un fichero XML.

8. El nodo raíz de un documento XML coincide con el ejemplar del mismo:

* Sí.
* No.

9. Indica cuáles de los siguientes elementos de un documento XML pueden ser nodos del mismo:

* Atributos.
* Comentarios.
* Etiquetas.
* Texto.

10. Las rutas de localización:

* Devuelven todos los nodos de un documento XML que verifican una condición dada.
* Devuelven todos los nodos de un árbol XML que verifican una condición dada.
* El primer nodo que se encuentra y cumple una condición dada.
* No pueden devolver valores de atributos.

Respuestas

1-b

2-d

3-a

4-a-b-c-

5-b y d

6: 3214

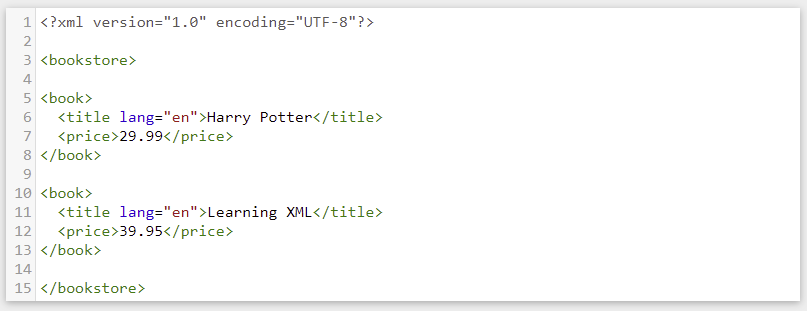
7: d

8: b

9: a b d

10.b

# w3schools



XPath usa expresiones de ruta para seleccionar nodos en un documento XML. El nodo se selecciona siguiendo una ruta o pasos. Las expresiones de ruta más útiles se enumeran a continuación:

|  |  |
| --- | --- |
| Expression | Description |
| *nodename* | Selects all nodes with the name "*nodename*" |
| / | Selects from the root node |
| // | Selects nodes in the document from the current node that match the selection no matter where they are |
| . | Selects the current node |
| .. | Selects the parent of the current node |
| @ | Selects attributes |

En la siguiente tabla, hemos enumerado algunas expresiones de ruta y el resultado de las expresiones

|  |  |
| --- | --- |
| Path Expression | Result |
| bookstore | Selects all nodes with the name "bookstore" |
| /bookstore | Selects the root element bookstore  Note: If the path starts with a slash ( / ) it always represents an absolute path to an element! |
| bookstore/book | Selects all book elements that are children of bookstore |
| //book | Selects all book elements no matter where they are in the document |
| bookstore//book | Selects all book elements that are descendant of the bookstore element, no matter where they are under the bookstore element |
| //@lang | Selects all attributes that are named lang |

## Predicados

Los predicados se utilizan para encontrar un nodo específico o un nodo que contiene un valor específico.

Los predicados siempre se incluyen entre corchetes.

En la siguiente tabla, hemos enumerado algunas expresiones de ruta con predicados y el resultado de las expresiones:

|  |  |
| --- | --- |
| Path Expression | Result |
| /bookstore/book[1] | Selects the first book element that is the child of the bookstore element.  Note: In IE 5,6,7,8,9 first node is[0], but according to W3C, it is [1]. To solve this problem in IE, set the SelectionLanguage to XPath:  *In JavaScript: xml*.setProperty("SelectionLanguage","XPath"); |
| /bookstore/book[last()] | Selects the last book element that is the child of the bookstore element |
| /bookstore/book[last()-1] | Selects the last but one book element that is the child of the bookstore element |
| /bookstore/book[position()<3] | Selects the first two book elements that are children of the bookstore element |
| //title[@lang] | Selects all the title elements that have an attribute named lang |
| //title[@lang='en'] | Selects all the title elements that have a "lang" attribute with a value of "en" |
| /bookstore/book[price>35.00] | Selects all the book elements of the bookstore element that have a price element with a value greater than 35.00 |
| /bookstore/book[price>35.00]/title | Selects all the title elements of the book elements of the bookstore element that have a price element with a value greater than 35.00 |

## Seleccionar nodos desconocidos

Los comodines XPath se pueden utilizar para seleccionar nodos XML desconocidos.

|  |  |
| --- | --- |
| Wildcard | Description |
| \* | Matches any element node |
| @\* | Matches any attribute node |
| node() | Matches any node of any kind |

En la siguiente tabla, hemos enumerado algunas expresiones de ruta y el resultado de las expresiones:

|  |  |
| --- | --- |
| Path Expression | Result |
| /bookstore/\* | Selects all the child element nodes of the bookstore element |
| //\* | Selects all elements in the document |
| //title[@\*] | Selects all title elements which have at least one attribute of any kind |

## Seleccionar varios caminos

Utilizando el | operador en una expresión XPath puede seleccionar varias rutas.

En la siguiente tabla, hemos enumerado algunas expresiones de ruta y el resultado de las expresiones:

|  |  |
| --- | --- |
| Path Expression | Result |
| //book/title | //book/price | Selects all the title AND price elements of all book elements |
| //title | //price | Selects all the title AND price elements in the document |
| /bookstore/book/title | //price | Selects all the title elements of the book element of the bookstore element AND all the price elements in the document |

## Ejes XPath

Un eje representa una relación con el nodo de contexto (actual) y se utiliza para ubicar nodos en relación con ese nodo en el árbol.

|  |  |
| --- | --- |
| AxisName | Result |
| ancestor | Selects all ancestors (parent, grandparent, etc.) of the current node |
| ancestor-or-self | Selects all ancestors (parent, grandparent, etc.) of the current node and the current node itself |
| attribute | Selects all attributes of the current node |
| child | Selects all children of the current node |
| descendant | Selects all descendants (children, grandchildren, etc.) of the current node |
| descendant-or-self | Selects all descendants (children, grandchildren, etc.) of the current node and the current node itself |
| following | Selects everything in the document after the closing tag of the current node |
| following-sibling | Selects all siblings after the current node |
| namespace | Selects all namespace nodes of the current node |
| parent | Selects the parent of the current node |
| preceding | Selects all nodes that appear before the current node in the document, except ancestors, attribute nodes and namespace nodes |
| preceding-sibling | Selects all siblings before the current node |
| self | Selects the current node |

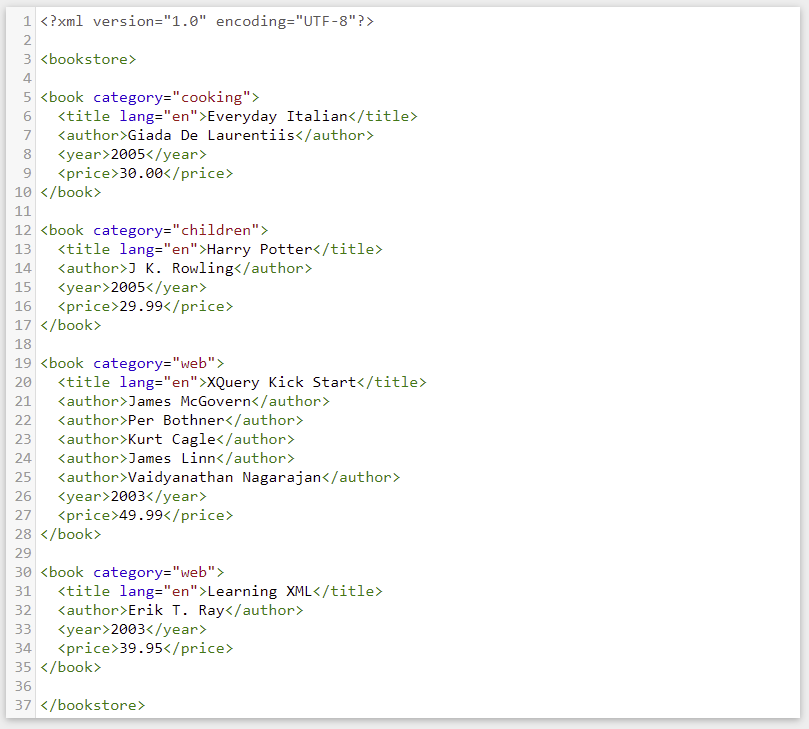
### Ejemplos

|  |  |
| --- | --- |
| Example | Result |
| child::book | Selects all book nodes that are children of the current node |
| attribute::lang | Selects the lang attribute of the current node |
| child::\* | Selects all element children of the current node |
| attribute::\* | Selects all attributes of the current node |
| child::text() | Selects all text node children of the current node |
| child::node() | Selects all children of the current node |
| descendant::book | Selects all book descendants of the current node |
| ancestor::book | Selects all book ancestors of the current node |
| ancestor-or-self::book | Selects all book ancestors of the current node - and the current as well if it is a book node |
| child::\*/child::price | Selects all price grandchildren of the current node |

## Operadores XPath

A continuación se muestra una lista de los operadores que se pueden usar en expresiones XPath:

|  |  |  |
| --- | --- | --- |
| Operator | Description | Example |
| | | Computes two node-sets | //book | //cd |
| + | Addition | 6 + 4 |
| - | Subtraction | 6 - 4 |
| \* | Multiplication | 6 \* 4 |
| div | Division | 8 div 4 |
| = | Equal | price=9.80 |
| != | Not equal | price!=9.80 |
| < | Less than | price<9.80 |
| <= | Less than or equal to | price<=9.80 |
| > | Greater than | price>9.80 |
| >= | Greater than or equal to | price>=9.80 |
| or | or | price=9.80 or price=9.70 |
| and | and | price>9.00 and price<9.90 |
| mod | Modulus (division remainder) | 5 mod 2 |



## Ejercicios

Seleccionar todos los títulos

/bookstore/book/title

Seleccione el título del primer libro

/bookstore/book[1]/title

Seleccione todos los precios

/bookstore/book/price[text()]

Seleccionar nodos de precio con precio> 35

/bookstore/book[price>35]/price

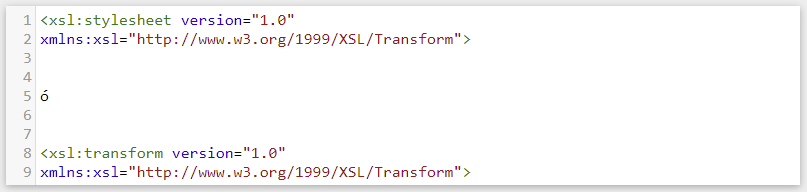
Seleccione los nodos de título con precio> 35

/bookstore/book[price>35]/title

# XSLT W3SCHOOLS



La forma correcta de declarar una hoja de estilo XSL de acuerdo con la Recomendación W3C XSLT es:



## El elemento <xsl: template>

El elemento <xsl: template> se utiliza para crear plantillas.

El atributo de coincidencia se utiliza para asociar una plantilla con un elemento XML. El atributo de coincidencia también se puede utilizar para definir una plantilla para todo el documento XML. El valor del atributo de coincidencia es una expresión XPath (es decir, match = "/" define todo el documento).



Dado que una hoja de estilo XSL es un documento XML, siempre comienza con la declaración XML: <? Xml version = "1.0" encoding = "UTF-8"?> .

El siguiente elemento, <xsl: stylesheet> , define que este documento es un documento de hoja de estilo XSLT (junto con el número de versión y los atributos del espacio de nombres XSLT).

El elemento <xsl: template> define una plantilla. El atributo match = "/" asocia la plantilla con la raíz del documento de origen XML.

El contenido dentro del elemento <xsl: template> define algo de HTML para escribir en la salida.

Las dos últimas líneas definen el final de la plantilla y el final de la hoja de estilo.

El resultado de este ejemplo fue un poco decepcionante, porque no se copiaron datos del documento XML a la salida. En el próximo capítulo, aprenderá a usar el elemento <xsl: value-of> para seleccionar valores de los elementos XML.

## El elemento <xsl: value-of>

El elemento <xsl: value-of> se puede usar para extraer el valor de un elemento XML y agregarlo al flujo de salida de la transformación:



El atributo de selección , en el ejemplo anterior, contiene una expresión XPath. Una expresión XPath funciona como navegar por un sistema de archivos; una barra inclinada (/) selecciona subdirectorios.

El resultado del ejemplo anterior fue un poco decepcionante; solo se copió una línea de datos del documento XML a la salida. En el próximo capítulo aprenderá a usar el elemento <xsl: for-each> para recorrer los elementos XML y mostrar todos los registros.

## El elemento <xsl: for-each>

El elemento XSL <xsl: for-each> se puede utilizar para seleccionar cada elemento XML de un conjunto de nodos especificado:



El valor del atributo de selección es una expresión XPath. Una expresión XPath funciona como navegar por un sistema de archivos; donde una barra inclinada (/) selecciona subdirectorios.

Filtrar la salida

También podemos filtrar la salida del archivo XML agregando un criterio al atributo de selección en el elemento <xsl: for-each>.

<xsl: for-each select = "catálogo / cd [artista = 'Bob Dylan']">

Los operadores de filtros legales son:

* = (igual)
* ! = (no igual)
* & lt; menos que
* & gt; mas grande que

## Dónde poner la información de clasificación

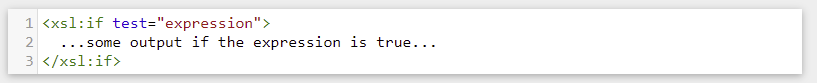
Para ordenar la salida, simplemente agregue un elemento <xsl: sort> dentro del elemento <xsl: for-each> en el archivo XSL:



El atributo de selección indica qué elemento XML ordenar.

## El elemento <xsl: if>

Para poner una prueba if condicional contra el contenido del archivo XML, agregue un elemento <xsl: if> al documento XSL.



### Dónde poner el elemento <xsl: if>

Para agregar una prueba condicional, agregue el elemento <xsl: if> dentro del elemento <xsl: for-each> en el archivo XSL:

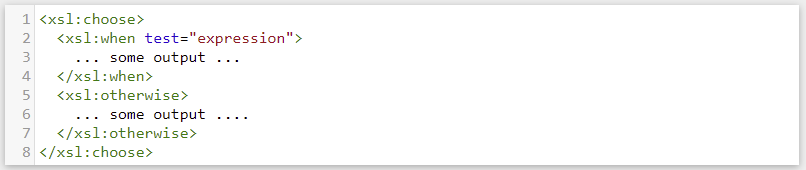


El valor del atributo de prueba requerido contiene la expresión que se va a evaluar.

El código anterior solo generará el título y los elementos del artista de los CD que tengan un precio superior a 10.

## El elemento <xsl: choose>

Sintaxis



### Dónde poner la condición de elección

Para insertar una prueba condicional múltiple contra el archivo XML, agregue los elementos <xsl: choose>, <xsl: when> y <xsl: else> al archivo XSL:



Otro ejemplo

Aquí hay otro ejemplo que contiene dos elementos <xsl: when>:



## El elemento <xsl: apply-templates>

El elemento <xsl: apply-templates> aplica una plantilla al elemento actual oa los nodos secundarios del elemento actual.

Si agregamos un atributo "select" al elemento <xsl: apply-templates>, procesará solo los elementos secundarios que coincidan con el valor del atributo. Podemos usar el atributo "seleccionar" para especificar en qué orden se procesarán los nodos secundarios.

Mire la siguiente hoja de estilo XSL:



### Una solución de JavaScript

En los capítulos anteriores hemos explicado cómo se puede utilizar XSLT para transformar un documento de XML a XHTML. Hicimos esto agregando una hoja de estilo XSL al archivo XML y dejamos que el navegador hiciera la transformación.

Incluso si esto funciona bien, no siempre es deseable incluir una referencia de hoja de estilo en un archivo XML (por ejemplo, no funcionará en un navegador que no sea compatible con XSLT).

Una solución más versátil sería utilizar JavaScript para realizar la transformación.

Al usar JavaScript, podemos:

hacer pruebas específicas del navegador

utilizar diferentes hojas de estilo de acuerdo con el navegador y las necesidades del usuario

¡Esa es la belleza de XSLT! Uno de los objetivos del diseño de XSLT era posibilitar la transformación de datos de un formato a otro, admitiendo diferentes navegadores y diferentes necesidades de los usuarios.